Kosmic Worlds V1 Technical Appendix

Egocentric Video World Model

Shaan Vats

Kosmic Worlds

shaan@kosmicworlds.com

October 2026

Abstract

We present Kosmic Worlds V1, a 1.6-billion-parameter egocentric video world model trained on approximately 3,600 hours of video from the Ego4D dataset [1]. Kosmic Worlds V1 operates in a compressed latent token space using the NVIDIA Cosmos video tokenizer [2] and learns an autoregressive spatiotemporal transformer over discrete latent video tokens. Generation can be conditioned on text, images, videos, and a learned latent action representation. V1 is designed to generate short-horizon worlds from text, image, or video inputs, with the long-term goal of building controllable, long-horizon world models for robotics, autonomous systems, synthetic data generation, games, and interactive simulation. This technical appendix describes the architecture, tokenization pipeline, training objective, distributed training setup, inference path, quantitative results, qualitative behavior, and observed limitations. Kosmic Worlds V1 is an early foundation system rather than a finished product. It establishes an end-to-end technical base for world model development at meaningful scale under limited compute and resources.

1 Introduction

Kosmic Worlds V1 was built to test whether a private, end-to-end world modeling stack could be trained at meaningful scale with limited compute. The stack includes dataset preprocessing, video tokenization, a custom spatiotemporal transformer, latent action conditioning, multi-GPU distributed training, checkpointing, evaluation, and an inference server.

The model contains approximately 1.6 billion parameters and was trained on approximately 3,600 hours of video. Video is represented as compressed latent tokens rather than pixels, allowing the dynamics model to learn spatial and temporal structure while drastically reducing the sequence length compared to direct pixel-based generation. Kosmic Worlds V1 processed 101 billion latent tokens, equivalent to 1.93 passes over the tokenized dataset.

The primary technical milestone is a reliable end-to-end training and inference pipeline. The model

was trained across eight NVIDIA B200 GPUs for approximately 125 hours. The training system supports checkpoint save and resume, optimizer and scheduler restoration, periodic evaluation, metric extraction, and text, image, or video conditioned generation. These results reduce technical uncertainty around training and operating larger and more capable Kosmic Worlds models.

Kosmic Worlds V1 remains limited. Current generations can be blurry, action conditioning is inconsistent, and long rollouts degrade. These are material limitations of an early model trained with constrained compute and imperfect data. The purpose of V1 is not to represent the final product, but to establish the technical base for Kosmic Worlds V2 and subsequent systems.

2 System Overview

Kosmic Worlds V1 is a latent video world modeling pipeline. Instead of generating pixels directly, the system converts video into a compressed sequence of discrete latent tokens using the NVIDIA Cosmos tokenizer [2]. A spatiotemporal transformer, based on the causal Transformer architecture [3], is then trained to model the dynamics of these token sequences autoregressively. At inference time, generated token sequences are decoded back into RGB video frames.

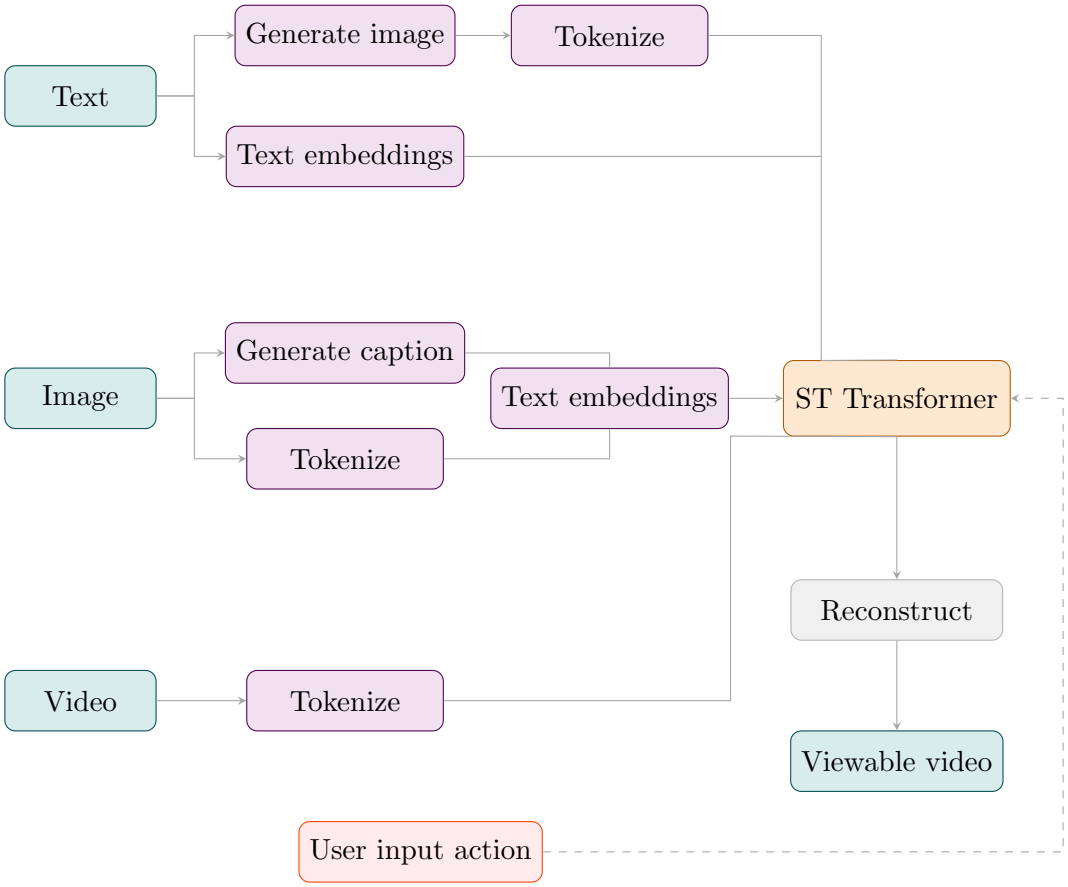


Figure 1: Kosmic Worlds V1 inference pipeline.

Kosmic Worlds V1 supports multiple input modalities. Generation can begin from text, an image used as the initial frame, or an input video. For text-to-video generation, Qwen2-1.5B-Instruct [14] expands the prompt into a detailed scene description, from which FLUX.1 [schnell] [15] generates the initial frame. Text conditioning is computed from the original prompt with a CLIP ViT-L/14 text encoder [4]. Image inputs are used directly as the initial frame and captioned with BLIP-2 [5], and the caption is encoded with CLIP. Video inputs are tokenized directly and currently carry no text conditioning; video-level captioning is planned for future models.

During training, each video sample is represented as a sequence of latent video tokens. The model receives previous latent tokens as context and learns to predict subsequent tokens autoregressively. The primary training objective is cross-entropy over the ground-truth target tokens, with an auxiliary loss for the latent action model.

The design separates video compression, latent dynamics modeling, and video reconstruction into distinct stages. The pretrained tokenizer compresses visual inputs into discrete latent representations. The spatiotemporal transformer models spatial and temporal dependencies over those representations. The inference system combines these stages to produce viewable RGB video.

3 Model Architecture

Kosmic Worlds V1 uses an autoregressive spatiotemporal Transformer [3] trained over discrete latent video tokens. The model does not directly predict RGB frames. Instead, it compresses each input video or image into latent tokens and predicts future tokens from prior visual context, text embeddings, and optional latent action conditioning.

Let $z_{1:T}$ denote a sequence of latent frames, where each frame $z_t \in \{0, \ldots, 63{,}999\}^P$ is a grid of $P = 32 \times 32 = 1{,}024$ discrete tokens, and let c denote text conditioning and a the latent action conditioning signal. The model is autoregressive over frames, with all tokens of a frame predicted in parallel:

$$p_\theta(z_{1:T} \mid c, a) = \prod_{t=1}^{T} \prod_{i=1}^{P} p_\theta(z_{t,i} \mid z_{<t}, c, a).$$

3.1 Spatiotemporal Transformer (Dynamics Model)

The Kosmic Worlds V1 dynamics model is a 1.6-billion-parameter causal spatiotemporal transformer. Each predicted latent token attends only to prior context (previous frames). Because every video timestep contains a spatial grid of latent tokens, the model must represent both frame-level spatial structure and temporal structure across frames.

Parameter	Value
Approximate parameter count	1.6B
Transformer layers	32
Model width	1512
Query heads	21
Key/value heads	3
Head dimension	72
MLP hidden dimension	6048
Tokenizer vocabulary size	64,000
Text conditioning dimension	768
Cross-attention heads	12
Latent action embedding dimension	32
Latent action code count	16

Table 1: Kosmic Worlds V1 model configuration.

The model configuration balances depth, representation width, attention capacity, and feed-forward capacity within the available compute budget. Parameters are distributed primarily across attention projections and feed-forward blocks. The transformer uses three-dimensional rotary positional embeddings derived from RoPE [6], factorized [13, 12] grouped-query attention [7], RMS normalization [8], and SwiGLU feed-forward blocks [17]. These components preserve causal generation while allowing each predicted token to depend on spatial context, temporal history, language conditioning, and optional action conditioning.

3.2 Text Conditioning

Text conditioning guides generation through cross-attention. Input text is encoded using CLIP ViT-L/14 [4], producing a sequence of language embeddings that the video tokens attend to through gated cross-attention in every block (Section 3.4).

3.3 Action Conditioning

Kosmic Worlds V1 includes a latent action conditioning pathway intended to provide a compact control signal to the spatiotemporal transformer. The latent action model receives consecutive latent frames, estimates the transition between them, and quantizes that transition into one of 16 learned action codes using a vector-quantized representation [9] with an exponential-moving-average codebook. At inference, an action code can be supplied directly. The selected code is embedded and added to the input of the dynamics model (Section 3.4).

Action conditioning is not consistently reliable in V1. Some action codes preserve coherent scenes,

while others accelerate degradation or produce little visible change. Consequently, the current results should be interpreted as evidence of a functioning action-conditioning pathway, not as evidence of robust semantic control. V2 replaces this pathway with physically grounded actions (Section 9).

3.4 Transformer Block

Let $h_{t,i} \in \mathbb{R}^d$, $d = 1512$, denote the hidden state of token i in latent frame t, and let $\mathrm{RMS}(\cdot)$ denote RMS normalization [8].

3.4.1 Input Embedding

Each token is embedded and summed with a gated embedding of the frame's action code,

$$h_{t,i}^{(0)} = E_z[z_{t,i}] + \sigma(\gamma_a)\, E_a[a_t],$$

where $\sigma(\gamma_a)$ is a learned scalar gate. The action embedding is shared across all positions in the frame.

3.4.2 Attention

All self-attention uses grouped-query attention with 21 query heads sharing 3 key/value heads, head dimension $d_h = 72$, query-key normalization [18], and three-dimensional rotary position embeddings $\mathrm{RoPE}(\cdot)$ [6] with 24 dimensions per axis:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{\mathrm{RoPE}(\mathrm{RMS}(Q))\,\mathrm{RoPE}(\mathrm{RMS}(K))^\top}{\sqrt{d_h}} + M\right) V.$$

Spatial attention (SA) operates over the tokens of a single frame with no mask, since all tokens of a frame are predicted jointly. Temporal attention (TA) operates over the same position across frames with a causal mask, $M_{ts} = 0$ for $s \le t$ and $-\infty$ otherwise.

3.4.3 Block Structure

Each of the 32 blocks, indexed by $\ell = 0, \ldots, 31$, maps $h^{(\ell)}$ to $h^{(\ell+1)}$ by applying factorized self-attention, gated cross-attention to the CLIP text embeddings c [19], and a SwiGLU feed-forward network [17]:

$$\begin{aligned} u &= h^{(\ell)} + \mathrm{TA}\big(\mathrm{SA}(\mathrm{RMS}(h^{(\ell)}))\big), \\ v &= u + \sigma(\gamma_\ell)\,\mathrm{CA}\big(u,\, \mathrm{RMS}(c)\big), \\ h^{(\ell+1)} &= v + W_3\big(\mathrm{SiLU}(W_1\mathrm{RMS}(v)) \odot W_2\mathrm{RMS}(v)\big), \end{aligned}$$

where CA is cross-attention with 12 heads, $\sigma(\gamma_\ell)$ is a learned per-layer gate, and the feed-forward network has hidden width 6048. Cross-attention is skipped when no text conditioning is provided.

3.4.4 Output Head

After the final block, a linear head maps each position to a distribution over the next frame's token at that position:

$$p_\theta(z_{t+1,i} \mid z_{\le t}, c, a) = \mathrm{softmax}\big(W_{\mathrm{o}}\,\mathrm{RMS}(h_{t,i}^{(32)})\big), \qquad W_{\mathrm{o}} \in \mathbb{R}^{64{,}000 \times d}.$$

3.5 Training Objective

Training minimizes the negative log-likelihood of the target latent tokens, averaged over all predicted tokens:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{TP} \sum_{t=1}^{T} \sum_{i=1}^{P} \log p_\theta(z_{t,i} \mid z_{<t}, c, a).$$

The latent action model is trained jointly with a commitment loss, which keeps its continuous outputs close to their assigned codes, and an entropy term, which encourages use of the full codebook. Writing these as $\mathcal{L}_{\mathrm{LAM}}$, the total training loss is

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{LAM}}.$$

4 Data and Tokenization

Kosmic Worlds V1 was trained on approximately 3,600 hours of Ego4D video [1]. V1 preprocessing was intentionally minimal: the pipeline removed redacted or near-black segments. The dataset therefore provides large-scale real-world video coverage while retaining substantial variation in clip quality.

Before training, clips are converted into discrete latent tokens using NVIDIA's Cosmos Tokenizer DV4x8x8 [2], which compresses video by 8×8 spatially and approximately $4\times$ temporally. At 256×256 resolution, each latent frame is a 32×32 grid of tokens. Latents are discretized with finite scalar quantization [16] using levels $[8, 8, 8, 5, 5, 5]$, giving a vocabulary of $8^3 \cdot 5^3 = 64{,}000$ codes. Let $x_{1:F}$ denote an RGB video clip with F frames and $\mathcal{E}$ the tokenizer encoder. The encoded representation is

$$z_{1:T} = \mathcal{E}(x_{1:F}),$$

where $z_{1:T}$ is the resulting sequence of discrete latent video tokens. During training, the transformer predicts subsequent latent tokens while attending only to causally valid visual context and any available text or action conditioning. During inference, the same tokenizer interface encodes visual context and decodes generated latent tokens into RGB frames.

5 Training Setup

Kosmic Worlds V1 was trained with PyTorch DistributedDataParallel [10] across eight NVIDIA B200 GPUs. The run used mixed precision, fused AdamW optimization [11], gradient clipping, linear warmup followed by cosine learning-rate decay, periodic evaluation, and periodic checkpointing.

Training Parameter	**Value**
FPS	16
Clip Resolution	256px
Clip Length	8 seconds
Approx. Number of Clips	1.6M
Tokens per Clip	32,768
Tokens processed	101B latent tokens
Tokens per epoch	52.5B
Epochs completed	1.93
GPUs	8× NVIDIA B200
Global tokens per step	2,621,440
Average throughput	234k tokens/s
Batch size	10 per GPU
Optimizer	AdamW
Peak learning rate	3×10^{-4}
Minimum learning rate	3×10^{-5}
Warmup ratio	0.01
Weight decay	0.1
Gradient clipping	1.0
Evaluation interval	500M tokens
Checkpoint interval	500M tokens

Table 2: Kosmic Worlds V1 training configuration.

Kosmic Worlds V1 processed 101 billion latent video tokens, corresponding to 1.93 epochs over the tokenized dataset. Training and evaluation progress is tracked by latent tokens processed rather than wall-clock time, allowing comparisons across hardware configurations and batch sizes.

Checkpoints were saved every 500 million latent tokens. Each resumable checkpoint stores model weights, optimizer state, scheduler state, gradient-scaler state, and the cumulative token count. This allows a long run to continue from an intermediate checkpoint without restarting warmup, resetting the learning-rate schedule, or discarding optimizer history.

6 Metrics and Results

Kosmic Worlds V1 is evaluated using optimization metrics and generated videos. The primary tracked metrics are training cross-entropy loss, evaluation cross-entropy loss, training perplexity, evaluation perplexity, LAM perplexity, learning rate, and training throughput. Training and evaluation losses are measured over latent video tokens. Lower cross-entropy indicates that the model assigns greater probability to the correct target tokens.

Perplexity is the exponential of the cross-entropy loss and provides an equivalent view of predictive uncertainty over the latent vocabulary:

$$\mathrm{PPL} = \exp(\mathcal{L}_{\mathrm{CE}}).$$

Another tracked metric is latent action model (LAM) perplexity, which measures effective codebook utilization. Let K denote the number of latent action codes, p_j the empirical assignment probability of code j, and ϵ a small numerical constant. The model computes LAM perplexity as

$$\mathrm{PPL}_{\mathrm{LAM}} = \exp\left(-\sum_{j=1}^{K} p_j \log(p_j + \epsilon)\right).$$

If $\mathrm{PPL}_{\mathrm{LAM}} \approx K$, assignments are distributed broadly across the codebook. If $\mathrm{PPL}_{\mathrm{LAM}} \approx 1$, assignments have collapsed toward a single code. This metric describes code usage, not semantic controllability by itself.

6.1 Loss Curves

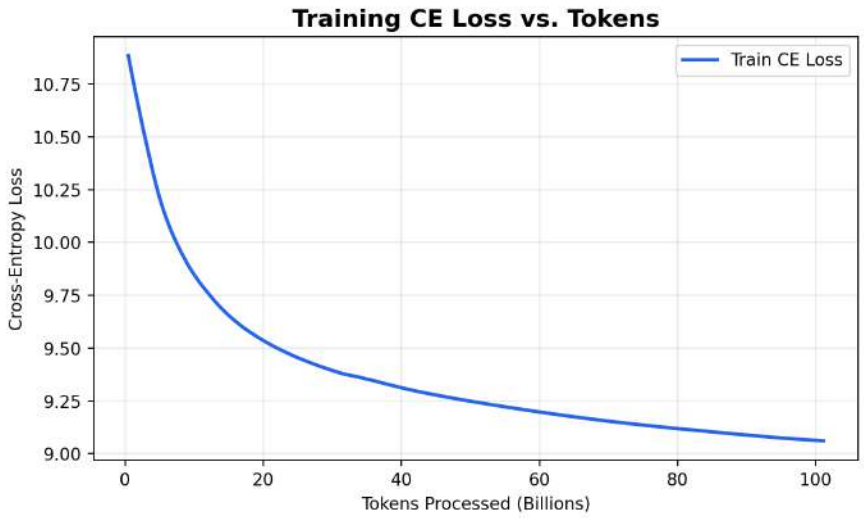


Figure 2: Training cross-entropy loss over latent tokens processed.

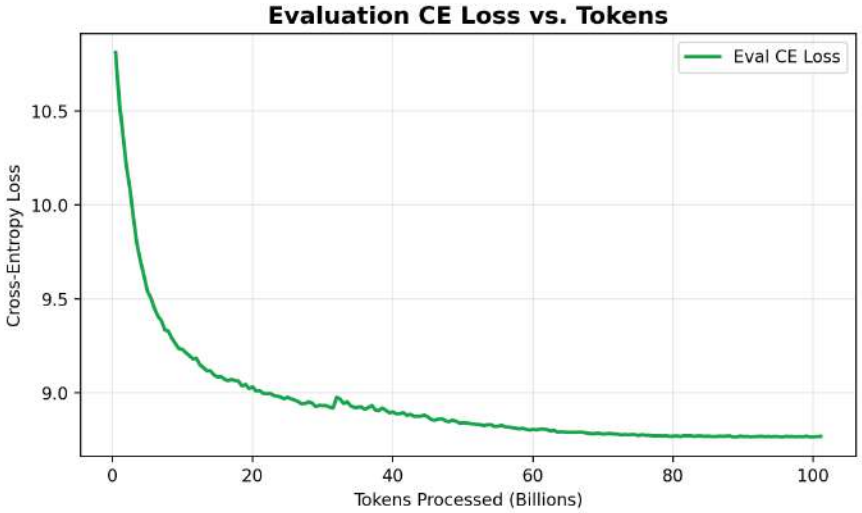


Figure 3: Evaluation cross-entropy loss over latent tokens processed.

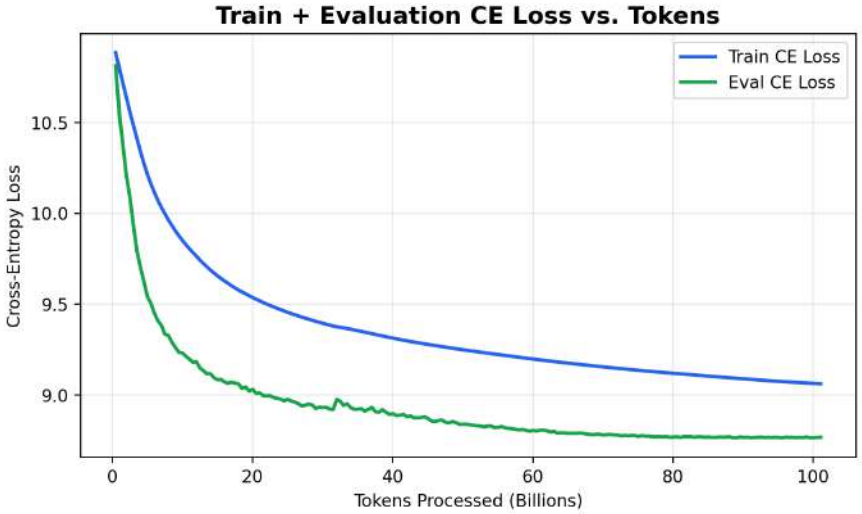


Figure 4: Training and evaluation cross-entropy loss overlay.

Training and evaluation loss decrease throughout the run, while the evaluation curve remains close to the training curve. This indicates continued improvement on the held-out evaluation split without a widening generalization gap. Improvements become smaller and noisier later in training as optimization approaches saturation.

The training-loss plot combines the original run and a resumed run. The visible change near 31.5 billion tokens reflects a reset of the running-average logger at the resume boundary, not an instantaneous change in model parameters. Evaluation loss is computed independently at each evaluation interval and does not contain this logging artifact.

6.2 Perplexity

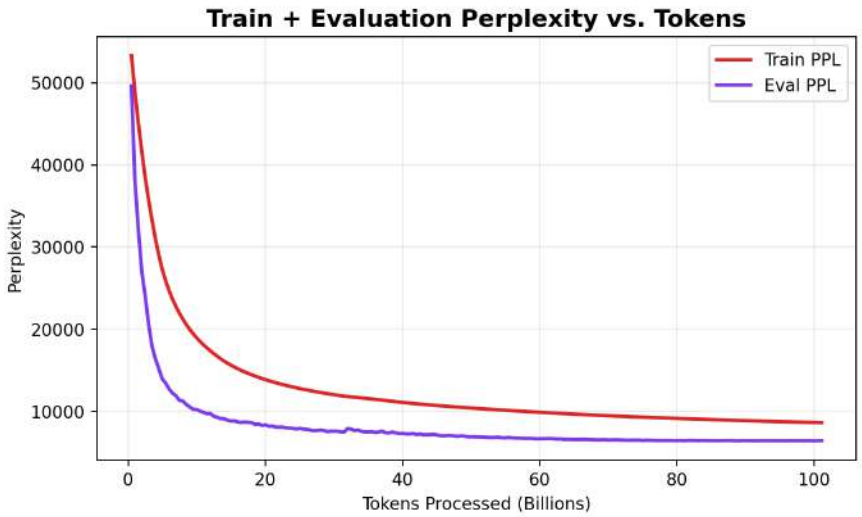


Figure 5: Training and evaluation perplexity overlay.

Perplexity follows the same trend as cross-entropy loss and provides an alternate view of predictive uncertainty over the latent vocabulary.

6.3 Latent Action Codebook Usage

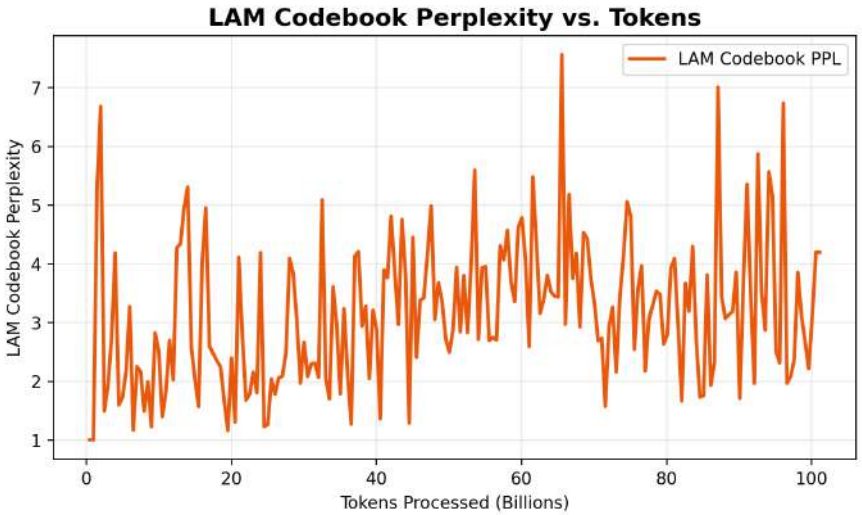


Figure 6: Latent action model perplexity over latent tokens processed.

Latent action model perplexity is tracked to monitor codebook utilization. A value near 1 indicates assignment collapse toward one code, whereas larger values indicate broader use of the 16-code

vocabulary. V1 exhibits fluctuating but nontrivial code usage. This metric does not establish that each code corresponds to a distinct, controllable semantic action; that requires separate intervention-based evaluation.

6.4 Optimization

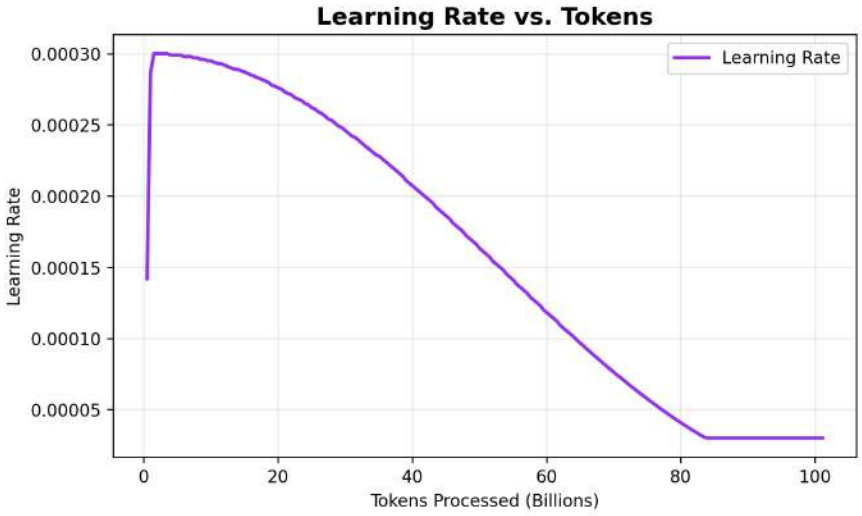


Figure 7: Learning rate schedule over latent tokens processed.

The learning-rate curve verifies the intended schedule: linear warmup followed by cosine decay toward the configured minimum learning rate of 3×10^{-5}.

6.5 Qualitative Results

Qualitative evaluation is performed by generating videos across different prompts, visual inputs, decoding settings, and action IDs. Kosmic Worlds V1 can produce recognizable environment structure and short-horizon scene persistence. The strongest results occur in comparatively simple environments with clear prompts and coherent initial frames or videos.

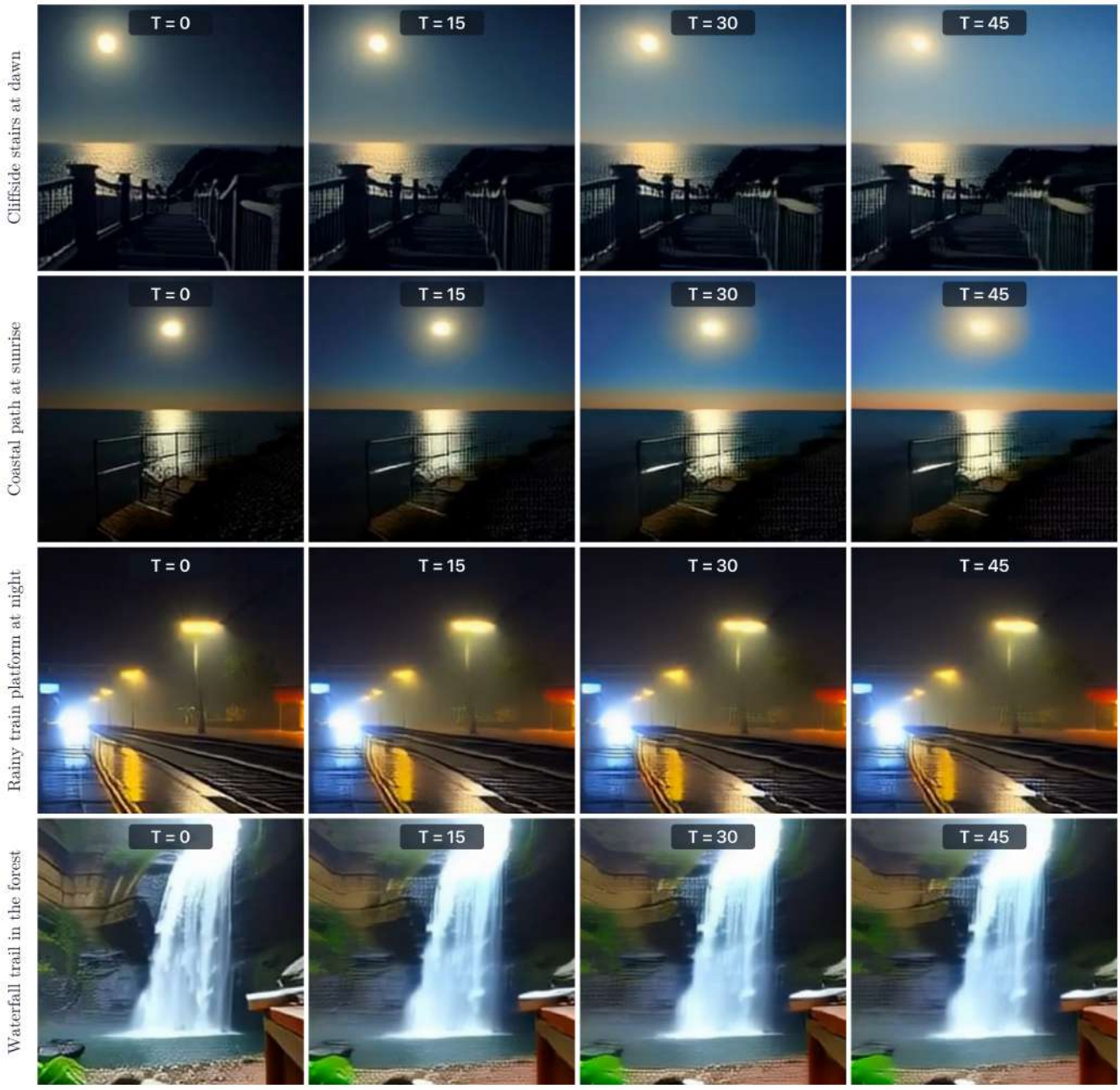


Figure 8: Text-conditioned generations from Kosmic Worlds V1. Each row shows one rollout at frame offsets $T \in \{0, 15, 30, 45\}$ (16 fps), conditioned on an initial frame generated by FLUX.1 [schnell] from the expanded prompt. All frames shown are generated by V1. Scene layout and lighting remain consistent over the rollout, and lighting evolves in line with the prompt in the first two rows, but camera and object motion are limited, consistent with the near-static behavior discussed in Section 9.

Longer generations degrade in predictable ways under autoregressive rollout. Sampling or prediction errors can alter future context and compound over time, eventually producing blurred or incoherent frames. Qualitative evaluation therefore includes both representative successful segments and recurring failure modes that motivate the design of V2.

7 Inference

Kosmic Worlds V1 supports text, image, and video conditioned inference. Generation occurs in latent space, one latent frame at a time. Given the visual context, the model predicts all 1,024 tokens of the next frame in parallel, either greedily or by sampling with temperature, top-k, and top-p filtering,

$$\hat{z}_{t+1,i} \sim p_\theta(z_{t+1,i} \mid \hat{z}_{\le t}, c, a),$$

appends the frame to the context, and repeats. The generated latents are decoded into RGB frames by the tokenizer decoder,

$$\hat{x}_{1:F} = \mathcal{D}(\hat{z}_{1:T}), \qquad F = 1 + 4(T-1),$$

so that 32 latent frames decode to 125 RGB frames. For longer rollouts, generation proceeds in a sliding window: the most recent 16 latent frames are retained as context, the next 16 are generated, and the overlap is removed before streaming.

8 Limitations

Kosmic Worlds V1 is an early foundation model with several important limitations. These limitations directly motivate the design of Kosmic Worlds V2 (Section 9).

First, generated videos can become blurry or lose spatial coherence over long rollouts. The strongest V1 generations can preserve scene layout, lighting, and overall color composition for approximately 3–5 seconds, but this duration is not consistent across prompts or decoding settings. A recurring failure mode is gradual smoothing and loss of structure. V2 is designed as an autoregressive and flow-matching hybrid to improve visual refinement and reduce rollout degradation.

Second, action conditioning is not yet reliable. Some action codes preserve coherent scenes longer than others, while some accelerate degradation or produce little visible change. Because decoding is stochastic in many experiments, code sweeps alone do not prove causal or semantic controllability. V2 moves from latent action codes to physically grounded actions.

Third, Kosmic Worlds V1 is constrained by the training-data distribution. Although large in scale, the minimally filtered dataset contains low-variance, low-motion, dark, repetitive, and visually degraded clips. These properties likely contribute to weak long-horizon motion and inconsistent object modeling. Future models will combine greater scale with substantially stronger filtering before tokenization.

Finally, Kosmic Worlds V1 generates natively at 256×256 resolution. Post-processing can improve presentation quality but cannot replace higher-resolution native modeling. Future systems will

tokenize and train on higher-resolution video.

9 From V1 to V2

Kosmic Worlds V1 was built to test the full world-modeling stack under limited compute. Beyond the system itself, it surfaced three lessons that shape the design of Kosmic Worlds V2.

A purely autoregressive objective compounds errors. Under autoregressive rollout, small prediction errors feed back into future context and accumulate over time. In V1, this appears as gradual blurring, loss of structure, and generations that settle into near-static scenes. V2 is designed as a hybrid that combines autoregressive generation with a flow-matching objective, preserving temporal causality while allowing each step to refine visual detail rather than commit to a single prediction.

Latent actions are unreliable for physical control. V1's learned latent action codes were sensitive to hyperparameter choices, and individual codes did not map consistently to identifiable changes in the scene. Codes learned without supervision are not physically grounded, so they cannot serve as control inputs for real systems. V2's fine-tuned variants are designed to be conditioned on continuous, physically grounded actions, such as steering wheel angle, throttle, and brake for driving. To obtain these labels at scale, we built Kosmic PACE, which recovers control inputs from egocentric driving video.

Short context limits long-horizon consistency. V1 was trained on 8-second clips, and its strongest generations preserve scene layout and lighting for only a few seconds. During long rollouts, each new segment is conditioned on only the most recent 16 latent frames, roughly four seconds of video, which closely matches the horizon over which V1's generations remain coherent. With no memory of the scene beyond this window, the model cannot keep objects, layout, and lighting consistent as a rollout grows longer. V2 will be trained on substantially longer sequences and is designed to maintain a persistent representation of the world across generation, so that scenes remain consistent over much longer horizons.

More on current work is available at kosmicworlds.com.

10 Conclusion

Kosmic Worlds V1 establishes the first working foundation for the company's video world-modeling system. It demonstrates that a private end-to-end pipeline for large-scale video tokenization, spatiotemporal transformer training, checkpointing, evaluation, and inference can be built and operated under limited compute and resources.

Kosmic Worlds V1 is not the final product, but it reduces technical uncertainty around building more capable Kosmic Worlds models. V1 shows measurable optimization progress, early video

world-modeling capability, and clear paths for improvement in data, architecture, controllability, visual quality, and long-horizon stability.

The next phase is to develop a higher-quality and more controllable Kosmic Worlds V2 system capable of producing longer, more stable, and more useful generated worlds.

11 Proprietary Implementation Note

Kosmic Worlds V1 is implemented in a private training and inference codebase. This appendix describes the system at an architectural and experimental level, but does not disclose full source code, implementation details, model weights, internal tools, or dataset access.

The purpose of this appendix is to provide enough technical detail to evaluate the system, its limitations, and its future direction without exposing implementation details that are core to the company's proprietary technology.

Additional technical details are available on request. Contact shaan@kosmicworlds.com.

References

[1] Kristen Grauman et al. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[2] NVIDIA Corporation. Cosmos Tokenizer: A suite of image and video neural tokenizers. Software repository, 2025. `https://github.com/NVIDIA/Cosmos-Tokenizer`.

[3] Ashish Vaswani et al. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

[4] Alec Radford et al. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[5] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[6] Jianlin Su et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024.

[7] Joshua Ainslie et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[8] Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems*, 2019.

[9] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*, 2017.

[10] Adam Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 2019.

[11] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

[12] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A Video Vision Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[13] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is Space-Time Attention All You Need for Video Understanding? In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[14] An Yang et al. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*, 2024.

[15] Black Forest Labs. FLUX.1 [schnell]. Model release, 2024. `https://huggingface.co/black-forest-labs/FLUX.1-schnell`.

[16] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite Scalar Quantization: VQ-VAE Made Simple. In *International Conference on Learning Representations*, 2024.

[17] Noam Shazeer. GLU Variants Improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[18] Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-Key Normalization for Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[19] Jean-Baptiste Alayrac et al. Flamingo: a Visual Language Model for Few-Shot Learning. In *Advances in Neural Information Processing Systems*, 2022.